Assignment 3

Advanced Computing Concepts for Engineering

Due April 6

The work that you turn in for this assignment must represent your individual effort. You are welcome to help your fellow students understand the material of the course and the meaning of the assignment questions, however, the answer that you submit must be created by you alone.

Q0. (a) [10] Design a readable context-free grammar for the 'tree expressions' over the alphabet {'(', ')', id, '.'}. Each tree expression is a finite sequence that fits one of the following rules.

- Each of id and num is a tree expression.
- A sequence of tree expressions surrounded by parentheses is a tree expression.
- A sequence of at least 2 tree expressions surrounded by parentheses, where the last is preceded by a dot, is a tree expression.

Here are some examples of tree expressions

( )
(id)
((id) (id id) id)
(( ) (id id) . id)

The following are not examples of tree expressions

(
id)
( )) (( )
(( ) . (id id) id)
(. (id id))

(b) [10] Using your grammar from part (a) trace the top-down predictive algorithm on the input

(( ) (id) . id) $

(c) [10] Design an LL(1) CFG for tree expressions. Calculate the selector sets for your grammar.

(d) [10] Design a recursive descent parser based on your grammar from part (c). Your parser should either return a tree representing the input or call 'error'. The tree is formed from 4 kinds of nodes

- NULL nodes are leaf nodes. I.e., they have no children. They correspond to the tree expression ()

- ID nodes are leaf nodes. They represent the expression id.
- CONS nodes have exactly 2 children. They represent expressions of the form (x . y) where x and y are (respectively) represented by the left and right children of the CONS node. That is (x . y) is represented by $\mathrm{CONS}(X, Y)$ where X and Y are trees representing x and y.

Furthermore a list without a dot such as

$$(x_0\ x_1\ \cdots\ x_n)$$

(with $n \geq 0$) will be represented by a tree

$$\mathrm{CONS}(X_0, \mathrm{CONS}(X_1, \cdots \mathrm{CONS}(X_n, \mathrm{NULL}) \cdots)) .$$

And list with a dot such as

$$(x_0\ x_1\ \cdots\ x_n\ .\ x_{n+1})$$

(with $n \geq 0$) will be represented by a tree

$$\mathrm{CONS}(X_0, \mathrm{CONS}(X_1, \cdots \mathrm{CONS}(X_n, X_{n+1}) \cdots)) .$$

(e) Optional: Code your parser in Java and test it. If he gets a chance, your instructor will provide some JUnit test cases.

Q1 [10]. In class we proved the unsolvability of the halting problem in two ways. We proved that a Turing Machine can not solve the halting problem for Turing Machines and that 'idealized Java' can not solve the halting problem for 'idealized Java'.

Show that the halting problem for 'idealized Java' can not be solved by a Turing Machine You may assume either that any Turing Machine can be translated into an equivalent idealized Java program, and conversely, any idealized Java program can be translated into an equivalent Turing Machine.